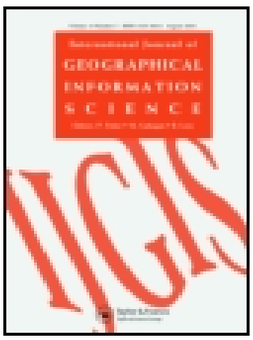
Machine learning for cross-gazetteer matching of natural features

Acheson, Elise ; Volpi, Michele ; Purves, Ross S

Abstract: Defining and identifying duplicate records in a dataset is a challenging task which grows more complex when the modeled entities themselves are hard to delineate. In the geospatial domain, it may not be clear where a mountain, stream, or valley ends and begins, a problem carried over when such entities are catalogued in gazetteers. In this paper, we take two gazetteers, GeoNames and SwissNames3D, and perform matching – identifying records in each that are about the same entity – across a sample of natural feature records. We first perform rule-based matching, establishing competitive results, then apply machine learning using Random Forests, a method well-suited to the matching task. We report on the performance of a wider array of matching features than has been previously studied, including domain-specific ones such as feature type, land cover class, and elevation. Our results show an increase in performance using machine learning over rules, with a notable performance gain from considering feature types, but negligible gains from other specialized matching features. We argue that future work in this area should strive to be more reproducible and report results on a realistic testing pipeline including candidate selection, feature extraction, and classification.

DOI: https://doi.org/10.1080/13658816.2019.1599123



International Journal of Geographical Information Science

ISSN: 1365-8816 (Print) 1362-3087 (Online) Journal homepage: https://www.tandfonline.com/loi/tgis20

Machine learning for cross-gazetteer matching of natural features

Elise Acheson, Michele Volpi & Ross S. Purves

To cite this article: Elise Acheson, Michele Volpi & Ross S. Purves (2019): Machine learning for cross-gazetteer matching of natural features, International Journal of Geographical Information Science, DOI: 10.1080/13658816.2019.1599123

To link to this article: https://doi.org/10.1080/13658816.2019.1599123

Published online: 22 Apr 2019.

INTERNATIONAL JOURNAL OF GEOGRAPHICAL INFORMATION SCIENCE
https://doi.org/10.1080/13658816.2019.1599123

Taylor & Francis
Taylor & Francis Group

RESEARCH ARTICLE

OPEN ACCESS Check for updates

Machine learning for cross-gazetteer matching of natural features

Elise Acheson[a], Michele Volpi[b] and Ross S. Purves[a]

[a]Department of Geography, University of Zurich, Zurich, Switzerland; [b]Swiss Data Science Center, ETH Zurich and EPFL Lausanne, Switzerland

ABSTRACT

Defining and identifying duplicate records in a dataset is a challenging task which grows more complex when the modeled entities themselves are hard to delineate. In the geospatial domain, it may not be clear where a mountain, stream, or valley ends and begins, a problem carried over when such entities are catalogued in gazetteers. In this paper, we take two gazetteers, GeoNames and SwissNames3D, and perform matching – identifying records in each that are about the same entity – across a sample of natural feature records. We first perform rule-based matching, establishing competitive results, then apply machine learning using Random Forests, a method well-suited to the matching task. We report on the performance of a wider array of matching features than has been previously studied, including domain-specific ones such as feature type, land cover class, and elevation. Our results show an increase in performance using machine learning over rules, with a notable performance gain from considering feature types, but negligible gains from other specialized matching features. We argue that future work in this area should strive to be more reproducible and report results on a realistic testing pipeline including candidate selection, feature extraction, and classification.

ARTICLE HISTORY
Received 30 April 2018
Accepted 20 March 2019

KEYWORDS
Gazetteer matching; record linking; random forest; natural features; feature types

1. Introduction

Defining and identifying duplicate records in datasets is an important and persistent problem in an age of increasing quantities of heterogeneous digital data, produced by diverse methods ranging from crowdsourcing to expert curation. Geographical datasets present unique challenges stemming from the vague nature of many geographical entities, the high degree of referent ambiguity in geographical names, and the varied categorization systems for entity types in this domain. These conceptual challenges manifest themselves in how geographical entities are catalogued in gazetteers, resources storing minimally geographical names, types, and geometries for a defined region of interest (Hill 2006).

Indeed, different gazetteers (or more broadly, geospatial databases) can have, for a particular region of interest, the same entity listed under different names (e.g. Lake

CONTACT Elise Acheson elise.acheson@geo.uzh.ch

Geneva vs. Le Léman), represented with different geometries (e.g. two different point centroids, or a point and a polygon), and assigned to different feature types, from different feature type hierarchies (e.g. mountain vs. Haupthuegel, German for 'main hill'). These representational issues are exacerbated when dealing with natural features[1] such as mountains and valleys, which may have vague or varying extents, and name matching is made more difficult when dealing with multilingual data. Furthermore, the number and type of entities listed in different gazetteers can vary greatly, leading to resources with orders of magnitude more records than others for a given area – that is, with higher spatial coverage (Ahlers 2013, Acheson *et al*. 2017a).

Duplicate detection is a well-studied problem that has cut across disciplinary boundaries, and is thus, somewhat ironically, associated with a variety of names, including deduplication, entity resolution, and record linking (as also noted by Elmagarmid *et al*. (2007)). In GIScience, when two or more geospatial datasets are being aligned, the process is widely referred to as matching (e.g. Walter and Fritsch 1999, Olteanu *et al*. 2006, McKenzie *et al*. 2014, Morana *et al*. 2014). In the specific context of placename resources (gazetteers), we thus refer to record linking as *gazetteer matching*. Gazetteer matching aims to identify records referring to the same real-world geographical entity, to then potentially merge or integrate these co-referential records while still presenting a coherent and consistent picture of the world.

Research on gazetteer matching has so far been relatively sparse, and standardized methodology, tools, and reference datasets have yet to be firmly established. Nonetheless, published methods often share approaches considering place names, geometries, and optionally feature types, to triage records and identify duplicates, either using hand-crafted rules (Fu *et al*. 2005, Hastings 2008, Smart *et al*. 2010, McKenzie *et al*. 2014) or machine learning (Sehgal *et al*. 2006, Zheng *et al*. 2010, Martins 2011, Gonçalves 2012, Gelernter *et al*. 2013). However, details about the datasets used are often hard to come by, as are the datasets themselves, and comparisons between rule-based methods and machine learning approaches are largely absent. Furthermore, the focus has been primarily on coarse granularity feature types such as cities (Martins 2011, Gonçalves 2012), and on finer granularity urban feature types such as points of interest (Zheng *et al*. 2010, Gelernter *et al*. 2013, McKenzie *et al*. 2014).

Natural features are a largely neglected subset of geographical records in matching tasks, despite being considered prototypical 'geographic features' by many (Smith and Mark 2003) and clearly presenting the aforementioned challenges of vagueness and diverse type classifications.

In this paper, we align a subset of natural feature records from a global gazetteer, GeoNames, to records from an authoritative gazetteer for our study area, Switzerland. Through this process, we make the following contributions:

- We use open datasets and, for a subset of gazetteer records, a publicly available annotated gold standard (Acheson *et al*. 2017b), thus enabling future work to be directly comparable.
- We implement machine learning methods for the matching task and compare these to rule-based methods.

- For a given machine learning model, we test a wider array of matching features than has been previously studied, considering domain-specific ones such as feature type, land cover class, and elevation.
- We provide a full pipeline evaluation and highlight the importance of creating a realistic testing pipeline to obtain representative performance. Indeed, we show how machine learning performance can be artificially affected by the choice of positive and negative record pairs.

In what follows, we motivate these contributions through a review of relevant work, particularly gazetteer matching and its methodological components in a rule-based and machine learning context (section 2). We then describe the two gazetteers used and our annotation process in section 3, followed by details of our rule-based matching methods and machine learning based methods in section 4. In section 5, we present the results of our approaches to automatically find matches between the two gazetteers. Our subsequent discussion centers around the many facets of matching that impact performance, and based on our detailed analysis we conclude with recommendations for future work.

2. Related work

Gazetteer matching is a special case of the widely studied problem of record linkage and part of the broader challenge of entity resolution (Elmagarmid *et al*. 2007, Costa 2011, Christen 2012). In gazetteer matching, the records to be linked represent geographical entities, rather than people, biomedical records, or web pages. The vast amounts of literature on record linkage, produced by various research communities, testify to the ubiquity of the problem, and to the added value of good solutions. As the need for gazetteer matching arises from heterogeneously catalogued data, we first briefly discuss the properties and uses of gazetteers, our data sources. We then focus on literature specifically concerned with gazetteer matching, and situate these works where appropriate within the broader context of entity resolution.

2.1. Gazetteers

Gazetteers, geographical datasets cataloguing named places alongside their feature types and geometries (Hill 2006, Berman *et al*. 2016), are important resources for linking unstructured textual content to geographical space. By providing explicit spatial representations (geometries such as points, lines, and polygons) for geographical references used in natural language (placenames, also known as toponyms), they can serve as 'glue' to explicitly spatialize textual data of various types, thus opening this data up for spatial analyses. Tasks that make use of gazetteers include detecting placenames in text, disambiguating and grounding placenames, and retrieving information about named places such as feature type, population, and containment relationships with other places or regions (Purves *et al*. 2007, Lieberman *et al*. 2010, Cooper and Gregory 2011, Adams *et al*. 2015).

Gazetteers are necessarily simplified versions of a more complex geographical reality, where continuous space is neatly carved up into objects and packaged into rows with attributes following a given schema. Data heterogeneity is all but unavoidable and can exist on several levels, including:

- **names**: places often have multiple names (whether spelling variants or across languages), may include fossilized type information within the name (e.g. Lake Placid), and the same name can be used for multiple entities, often in close proximity (e.g. Lake Placid the town and Lake Placid the lake) (Brunner and Purves 2008, Hastings 2008).
- **geometries**: the geometries representing named places can be different (e.g. two different points), of a different type (e.g. a point and a line), and of varying geometric complexity.
- **feature types**: each gazetteer can have its own feature type hierarchy used to classify real-world entities, and within a hierarchy, different types could be assigned to gazetteer records representing the same entity.

With the vast quantities of geotagged data available today online, particularly produced by non-experts and pushed to various social media or photo-sharing platforms, gazetteer production processes have evolved to include not just top-down, curated resources, but 'bottom-up' gazetteers from crowdsourced data (Popescu *et al*. 2008, Keßler *et al*. 2009, Gao *et al*. 2017). This means more resources exhibiting potentially high levels of heterogeneity and requiring both internal deduplication and robust record linking across datasets.

2.1.1. Feature type hierarchies

Categories of geographical entities are useful for communication and reasoning in everyday situations, such as when describing a hike 'through a **valley**' or 'up a **mountain**'. In the context of gazetteers, these categories are known as feature types, and feature type categorization systems are referred to varyingly as feature type hierarchies (our preferred term), schemes, thesauri, or ontologies, depending on their characteristics (e.g. see Janowicz and Keßler (2008)). Feature type hierarchies are one important way in which gazetteers vary, since these hierarchies tend to be idiosyncratic, with types appropriate for a gazetteer's area of coverage, language(s), and purpose (Hastings 2008, Janowicz and Keßler 2008).

Works dealing with aligning feature type hierarchies in a gazetteer matching context face a 'chicken or egg' problem: the feature types of records may be used as evidence that two records are about the same entity, and linked records may in turn be used to calculate feature type similarity. Taking a pragmatic approach, Brauner *et al*. (2007) calculate similarity measures between feature types in two gazetteers based on records deemed to be about the same entity, but offer few details on this record linking process, which relies heavily on geographic location alone. Hastings (2008) manually conflates different feature type hierarchies into one in order to then perform gazetteer matching. Smart *et al*. (2010) follow a similar path, developing a custom feature type ontology based on their gazetteer data sources and suitable for their tasks. In a machine learning based matching context, Sehgal *et al*. (2006) use annotated record pairs to calculate a static type similarity metric between feature type pairs, then used in classification. In a work concerned with type alignment rather than gazetteer matching, Zhu *et al*. (2016) calculate spatial statistics for records of a subset of feature types to derive 'spatial signatures' for each type, which they argue may complement existing type alignment methods.

2.2. Gazetteer matching

Entity resolution in general consists of a sequence of steps (Figure 1), usually comprising a *data preparation* step (establishing which fields are to be compared and cleaning and normalizing records), a *record linking* step (matching corresponding records), and potentially a *record fusion* step (merging/augmenting records deemed to be about the same entity) (Elmagarmid *et al*. 2007, Costa 2011). The second step, record linking, is the focus of this paper and, in the context of our work, we refer to it as gazetteer matching. Thus, gazetteer matching consists of linking pairs of gazetteer records which are thought to refer to the same real-world entity. A special case of the problem is deduplication, where duplicate records in a single gazetteer are identified and merged (Christen 2012). Cross-gazetteer matching, however, presents additional challenges compared to deduplication, including dealing with multiple feature type hierarchies (Janowicz and Keßler 2008), with structural heterogeneity such as different schemas (Elmagarmid *et al*. 2007), and with varying spatial coverage and balance between gazetteers (Acheson *et al*. 2017a).

Broadly, matching methods can be divided into rule-based (or distance-based) approaches and machine learning (or probabilistic) approaches (Elmagarmid *et al*. 2007). Rule-based approaches rely on either a series of binary rules for triaging records until only matches remain, or on setting weights manually to a set of distance (or similarity) measures to identify the most similar record(s) for each candidate. Rule-based methods can be considered a special case of distance-based methods, where distances are boolean (Elmagarmid *et al*. 2007). Machine learning approaches treat the matching problem as a supervised binary classification problem, outputting a 'match' or 'no match' label for each candidate pair of records. To train such a classifier, a set of matching record pairs (positive training examples) and non-matching pairs (negative training examples) are required. The key difference between rule-based and machine learning methods is thus whether decision boundaries between matches and non-matches are defined heuristically or in a data-driven fashion, with the latter explicitly requiring training data. Generally, machine learning methods are considered to offer more flexible and performant solutions, with the caveats of being more complex to implement and more time-consuming due to the need for annotated training data. Correspondingly, the main advantages of rule-based methods over machine learning methods are their simplicity and the reduced need for annotated data, since many rule-based methods only use data in evaluation, but not training (Elmagarmid *et al*. 2007). For gazetteer matching, this is an appealing advantage due to the paucity of annotated

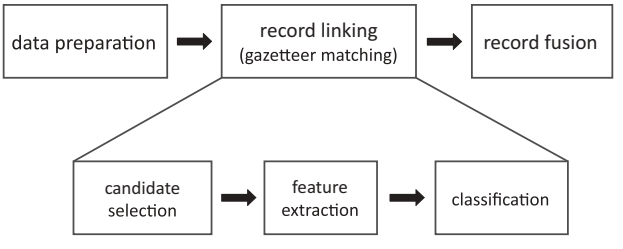


Figure 1. Entity resolution steps and sub-steps.

datasets, despite some recent efforts, including a benchmark building tool (Morana *et al*. 2014) with restrictive target datasets, and a public, but not cross-gazetteer, dataset (Gonçalves 2012).

2.2.1. Gazetteer matching steps

Gazetteer matching can itself be divided into steps (Figure 1). Whether one takes a rule-based or machine learning approach, the first step to finding matching records is to find match candidates, known as *candidate selection* (Zheng *et al*. 2010). Indeed, for any reasonably large dataset, considering every pair of records as a match candidate is infeasible and unnecessary: geographic records with locations that are far apart are improbable matches. This candidate selection step is also known as 'filtering' (Martins 2011), 'blocking' (Elmagarmid *et al*. 2007, Morana *et al*. 2014), or 'indexing' (Christen 2012). The general idea is, for every source record, to select from all target records only a subset of likely match candidates. For example, Zheng *et al*. (2010) select candidates using 'simple heuristics' including ignoring records with locations too far apart or with completely dissimilar names. In practice, candidate selection can consist of an initial coarse triage of target records, followed by a more careful selection of candidates via secondary filtering criteria or a ranked list. In one such case, Morana *et al*. (2014) use a feature-type specific point-radius method as an initial spatial filter alongside a type filter, then retain any candidate that shares a token with the name to match (e.g. New Amsterdam vs. New York), a pragmatic requirement for their monolingual points of interest (POI) data in a language which does not use compound nouns.

Once matching candidates are identified for a particular record, distance (or similarity) metrics can be calculated between each remaining pair of records. In the context of machine learning, these metrics are called 'matching features', and thus this step is known as *feature extraction* (Zheng *et al*. 2010). Pairwise metrics are typically calculated on place names, locations (geometries), and feature types (Sehgal *et al*. 2006, Zheng *et al*. 2010, Martins 2011). In particular, many algorithms exist which estimate name similarity, including character-based, token-based, and phonetic algorithms (Elmagarmid *et al*. 2007). Sehgal *et al*. (2006) find that the character-based Levenshtein distance (Levenshtein 1966) (also known as edit distance) is the best similarity metric for location names, but Zheng *et al*. (2010) argue that custom token-based metrics are more suitable for their POI and address data. Smart *et al*. (2010) use a combination of the Levenshtein distance, text normalisation, and the SoundEx phonetic algorithm. Martins (2011), extended in Gonçalves (2012), explore a wide variety of name similarity metrics for gazetteer record deduplication, including the Levenshtein, Jaro-Winkler, Monge-Elkan, Double Metaphone, and Jaccard algorithms. As for matching features on feature types, options vary based on whether types are from a single hierarchy and whether records can have more than one type assigned. With one hierarchy and multiple types per record, matching features can be derived based on the hierarchical distance between the types and on the overlap between assigned types (Zheng *et al*. 2010, Martins 2011). With multiple typing systems, options include manually aligning relevant types (Morana *et al*. 2014) or estimating type alignment based on annotated records (Sehgal *et al*. 2006).

After selecting candidate pairs and calculating matching features for each pair, a final *classification* step outputs a decision for each pair as to whether they form a match or not. Using rules, such a decision can be made by manually setting a threshold on an overall similarity score or on individual scores, whereas using machine learning, decisions are

probabilistic and depend on the algorithm and training data used to build a predictive model. Machine learning algorithms used for the gazetteer matching task have included Logistic Regression, Voted Perceptron (Neural Network), Support Vector Machines (SVMs), Decision Trees, and Random Forests (Sehgal *et al*. 2006, Zheng *et al*. 2010, Martins 2011, Gonçalves 2012). In particular, random forests have been found to outperform both decision trees and SVMs in a gazetteer deduplication context (Gonçalves 2012). Deep learning has not yet been used in gazetteer matching, but Santos *et al*. (2018) use deep learning in the form of a recurrent neural network to classify whether pairs of placenames are in fact alternate names for the same geographic entity, using the large GeoNames gazetteer as training data. However, deep learning requires very large annotated subsets for training, a property which is not likely to be satisfied in most heterogeneous gazetteer matching tasks.

2.2.2. Training and testing in machine learning based matching

A particular challenge in machine learning based gazetteer matching is the selection of training and test data. How does the size and composition of the training data impact classification performance? In particular, what ratio of matches (positive training examples) to non-matches (negative training examples) should the training data comprise and how should non-matches be selected? How should test data be selected and processed and will performance on this test data generalize to a wider dataset?

Sehgal *et al*. (2006) investigate in some detail how to choose non-matches for training in a gazetteer matching context, settling on a combination of random non-matching pairs and 'hard negatives', where hard negatives are non-matching records that have either highly similar names or highly similar locations. They choose the top k hard negatives per record, where k is optimized experimentally based on classification performance. Their highest performing algorithm (Logistic Regression) and training set composition consists of 30 negative training examples for every positive example. Martins (2011) and Gonçalves (2012) follow a similar procedure but opt for a 1:1 ratio of non-matches to matches, additionally specifying that half of their non-matches are selected randomly, while the other half comes from a ranked list of hard negatives for the whole collection. In Zheng *et al*. (2010), few details are given as to how non-matches were selected, but their overall dataset consists of a 1:1 ratio of matches and non-matches (800 each).

In addition, cross-validation is often the end point of a machine learning classification pipeline, with no independent, unseen test set. Since it is well known that unbalanced datasets and poorly chosen evaluation data can lead to wildly overoptimistic (or indeed pessimistic) evaluations in a wide range of contexts (Murphy 1996), we argue that there is a need for increased clarity in how training and testing data are chosen and processed in gazetteer matching.

3. Data

3.1. Application context

Our specific motivation for performing gazetteer matching is to aid in the georeferencing of Swiss alpine journal texts. These texts consist largely of descriptions of ski tours and hikes, and thus contain textual references to natural features such as mountains, valleys, and glaciers. Most of the natural features mentioned in these texts are located in

Switzerland, but some are in other mountainous parts of the world. Thus we use both SwissNames3D, an official placename resource for Switzerland, and GeoNames, an unofficial global resource. We wish to reconcile overlapping records for a cleaner georeferencing process, by linking GeoNames records to their SwissNames3D equivalent(s) in order to potentially increase recall while maintaining precision.

3.2. Gazetteers

SwissNames3D is an authoritative gazetteer of placenames in Switzerland, which is freely downloadable online.[2] A new edition is published annually, with the full country-wide update cycle taking 6 years. We downloaded the full dataset in February 2017. It contains over 300k records, each with a unique identifier, organized according to a Switzerland-specific feature type hierarchy, with lines and polygons available for a large subset of records depending on their types (e.g. streams are available as lines and valleys are available as polygons).

GeoNames is a widely used global gazetteer composed from a variety of sources including open geographical datasets and user-contributed data. Because of its global coverage and easy availability, GeoNames is very widely used, though it has also been recognised that its bottom-up production makes understanding data quality challenging (Ahlers 2013, Acheson *et al*. 2017a). We downloaded the freely available, daily updated GeoNames data for Switzerland on 20 July 2017.[3] It contains around 67k records for the country, organized according to a two-tiered, global feature type hierarchy, and with points available for all records in this free version.

3.3. Annotation

We manually prepared an annotated gold standard for a portion of records in GeoNames. Since SwissNames3D is an official resource and contains a much larger number of records than GeoNames for Switzerland, we assumed it to be more accurate and complete, thus better suited as our target resource for matching. We thus started with our source dataset GeoNames, and retained all feature types that we identified as representing natural (as opposed to human-made) geographic features and having at least 100 records in Switzerland. From the 8 types that met these criteria, we randomly selected 50 records of each type for annotation (see Table 1).

Table 1. Selected natural feature types from GeoNames, along with a representative type in SwissNames3D, and their counts.

GeoNames			SwissNames3D	
type	count	annotated	type	count
lake (LK)	1132	50	See	1263
glacier (GLCR)	806	50	Gletscher	854
stream (STM)	172	50	Fliessgewaesser	6603
peak (PK)	6557	50	Gipfel	2225
pass (PASS)	1785	50	Pass	2290
hill (HLL)	665	50	Huegel	1840
mountain (MT)	352	50	Haupthuegel	938
valley (VAL)	113	50	Tal	2260

For these 400 GeoNames records, one annotator per record tried to comprehensively find matches in SwissNames3D, including one-to-many and 'no match' cases. Any harder cases were then discussed among the four annotators, all graduate students in Geographic Information Systems, until an agreement was reached (as described in Acheson *et al*. 2017b). These annotated data provide us with training and evaluation data for matches, but not non-matches, in the gazetteer matching experiments which we now describe.

4. Methods

We implemented and compared rule-based matching and machine learning based matching. In terms of the overall entity resolution pipeline laid out in Figure 1, in both cases we performed the *data preparation* step manually, then focused on the core *record linking/gazetteer matching* step and its sub-steps. *Record fusion* (merging/augmenting linked records) was not performed for this work, but as SwissNames3D is an authoritative resource, fusion could consist of adding information from GeoNames which is not present in SwissNames3D, such as some alternate names. Data preparation included identifying which fields should be compared, and projecting the GeoNames latitude and longitude coordinates (WGS84) to Swiss coordinates (LV03) and vice-versa, to facilitate distance calculations. Additional work was required to deal with a peculiarity of SwissNames3D, which uses multiple records, each with the same table ID, for a single geographic entity when this entity has an official name in more than one official language of Switzerland. To get around this issue, we created a truly unique ID for each record, then ran our entire pipeline treating every record as unique, before reconstructing the original IDs for evaluation in order to not underestimate recall.

In the remainder of this section, we give an overview of our matching features, then present our rule-based matching methods. We then focus on our machine learning matching pipeline and how we implemented each step, and finally present our evaluation methods.

4.1. Matching features overview

Previous work has consistently used record names and geometries in gazetteer matching, firmly establishing their utility. Consequently, our rules and machine learning feature combinations always consider names, minimally as the Levenshtein distance between pairs of record names, and geometries, as the point-to-point distance between records (Vincenty 1975); the only exception is our rule-based random baseline which only considers names. As for feature types, their use and treatment is less consistent in the literature, with many derived matching features requiring a single feature type hierarchy. In this work we use feature types in two main ways: as an initial filter to limit the number of target records we consider during candidate selection, and as categorical features for machine learning during feature extraction. For candidate selection, we first established soft type alignments (Table 2) between the types of interest in our two different feature type hierarchies (a Switzerland-specific hierarchy with types in German, and a global hierarchy with types in English), using feature type metadata rather than our annotated data. However during feature extraction for machine learning,

Table 2. Soft type alignments used and type-specific distance thresholds.

GeoNames type	SwissNames3D types	Threshold (km)
mountain (MT)	Hauptgipfel, Gipfel, Huegel, Haupthuegel, Alpiner Gipfel, Grat, Huegelzug, Felskopf	5
hill (HLL)	Hauptgipfel, Gipfel, Huegel, Haupthuegel, Alpiner Gipfel, Grat, Huegelzug, Felskopf	5
peak (PK)	Hauptgipfel, Gipfel, Huegel, Haupthuegel, Alpiner Gipfel, Grat, Huegelzug, Felskopf	5
glacier (GLCR)	Gletscher, Alpiner Gipfel	5
pass (PASS)	Pass, Graben	5
lake (LK)	See, Seeteil	15
stream (STM)	Fliessgewaesser	15
valley (VAL)	Tal, Haupttal, Graben	15

we do not use these alignments, and instead encode the categorical feature type information as a set of binary features labelling the presence or absence of a given type, a common strategy known as 'one-hot encoding'. We also indirectly use feature types in one of our rule-based procedures to define type-specific geographical distance thresholds. In addition to names, geometries, and types, we also make use of elevation and land cover information as a way to represent properties of the natural environment, which may be useful in matching our natural feature records. A detailed list of all our matching features is given in 4.3.3.

4.2. Rule-based matching

We implemented rule-based methods to obtain competitive results while also testing the utility of matching features in a deterministic way. We tested the following rule-based matching procedures, in order of increasing complexity:

- **random-baseline**: find all exact name matches on the primary name for a given source record, then randomly choose one exact match as the match (no exact name matches means no match).
- **name-threshold**: find all exact name matches on the primary or any alternate name for a given source record, then from these, retain all target records within a fixed distance threshold (e.g. 5km) of the source record; here, the set of results can have 0, 1, or multiple matches per source record.
- **name-custom-threshold**: proceed as in *name-threshold* above, but this time use custom thresholds (see Table 2) specific to the feature type of the source record (c.f. Morana *et al*. 2014).
- **multi-threshold**: proceed as in *name-custom-threshold* above, but discard any target records above a threshold on land cover distance (as described in 4.3.3) or elevation.
- **linear-combination**: find all exact name matches on the primary or any alternate name as before, and additionally retain any target records of an aligned feature type (see Table 2) for each source record, then calculate edit distance (Levenshtein) and geographical distance for each pair. Combine these two distances in a weighted sum for a final score and keep any pairs with a score above an empirically derived threshold as a match (c.f. Smart *et al*. 2010).

Our rule-based procedures above combine matching features either by sequentially applying thresholds (*name-threshold, name-custom-threshold, multi-threshold*) or by including them in a linear combination (*linear-combination*). Since an important real-world advantage of rules over machine learning is simplicity, in terms of both interpretability and development time, we spent some time optimizing thresholds (for geographical distance and in *linear-combination*, for the overall score), but did so manually by considering sensible values for distance thresholds and by testing values at fixed intervals for the overall score. Furthermore, the more matching features are used, the more thresholds or weights there are to optimize overall, which incentivizes a sparser use of matching features. As a compromise between using a minimum number of matching features and having to use machine-learning-like techniques to combine a large number of features, we include a procedure (*multi-threshold*) which combines at least one matching feature from each of the five categories (names, geometries, types, elevation, and land cover).

4.3. Machine learning based matching

We frame machine learning based matching as a binary classification problem. Our aim is to build a model to infer whether a pair of records is a match or not, that is, whether they refer to the same real-world entity. To this end, we use a Random Forest classifier (Breiman 2001). We selected this classifier for several reasons. First, its simplicity: Random Forest is a non-linear, non-parametric classifier, which is intrinsically regularized by ensembling and not prone to overfitting. The key parameter choice in a random forest is the number of trees used in the ensemble: the larger the number, the less prone to overfitting the model is. We settled on 200 trees after finding that performance plateaued around this number. Second, as opposed to most probabilistic and distance-based classifiers (e.g. non-linear Support Vector Machines, Naive Bayes, etc.), input features do not have to be pre-processed to follow some particular distribution, nor do they have to be normalized. Random forests can also naturally handle categorical and continuous features jointly, a key advantage in our case to work with categorical features, such as feature types and land cover classes, alongside continuous features, such as geographical distance and elevation. As mentioned, categorical features are 'one-hot encoded', that is, M categories are encoded as M binary features labelling the presence or absence of a given category (e.g. category 4 out of 5 is thus encoded as [0,0,0,1,0], category 2 as [0,1,0,0,0] and so on). Finally, random forests have been successful in many applications, being better or at least on par with most non deep-learning classification algorithms.

In our implementation, we ask the random forest to infer whether a previously unseen pair of records (test data) is a match or not. To this end, the feature vector representing the pair is passed through every tree and the ensemble outputs a distribution over the labels. The final solution is given by taking the maximum-a-posteriori over the predicted posterior.

4.3.1. Pipeline overview

We built a machine learning processing pipeline (Figure 2) with the aim to approximate a realistic, large-scale matching scenario. First of all, we assume the dataset is too large to calculate matching features for every possible record pair, and thus the

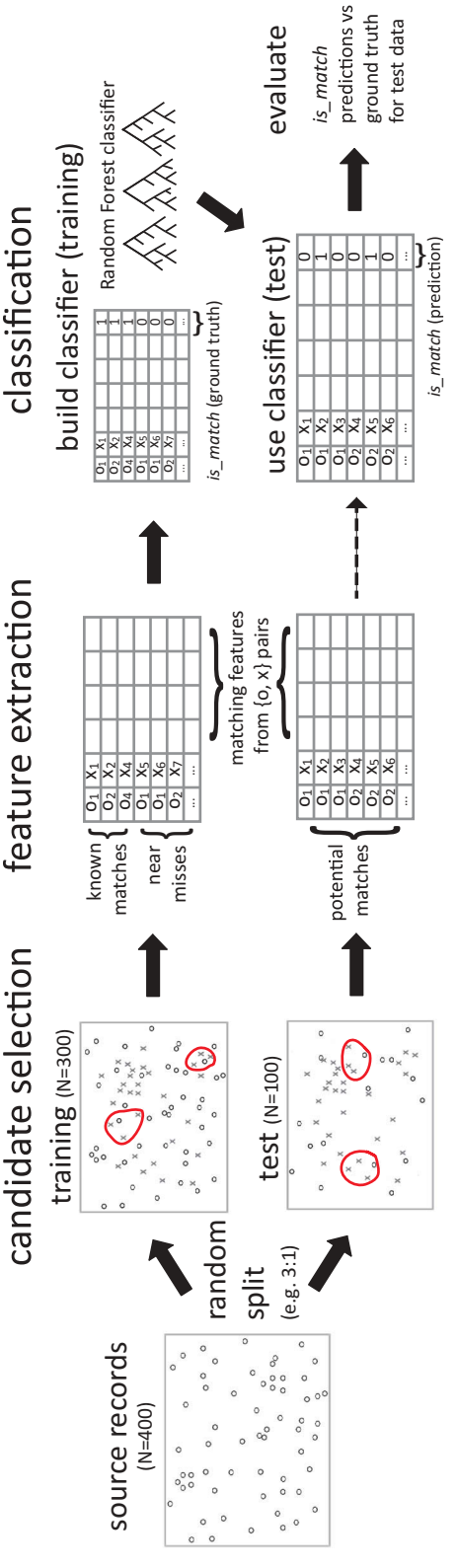


Figure 2. Machine learning pipeline overview.

first matching step must be to choose a subset of pairs, via candidate selection. Second, we assume that true positive pairs are not known in advance for the test set, as for truly unseen data, and thus positive matches in our test set must all be found through candidate selection. Indeed in our testing pipeline, we only classify pairs with target records retained at the candidate selection stage. In our training pipeline however, we ensure that the full set of annotated matching pairs for our source training records gets used, allowing our classifier to be optimised based on all of the information we have available. In order to process the training and test set in this slightly different manner, we split the source records at the beginning of a run, prior to candidate selection.

A single run of the pipeline takes a portion of source records through the training pipeline to train a classifier, then evaluates this classifier on the remaining, held-out, source records which go through the testing pipeline. As a compromise between maximizing the size of the training data and having a large enough test set, we opt to split the full set of source records ($N = 400$) randomly as 75% training and 25% test for our main runs. We perform 20 runs with this splitting procedure, each time testing our full set of feature combinations (section 4.3.4) on this particular split, to obtain not only mean values of precision, recall, and F1 per classifier, but also interquartile ranges (results in section 5.2). As we do not constrain the random split, the test sets vary in composition and difficulty, which is desirable to ensure robustness in a real-world scenario. We also create a fixed, feature-type-balanced test set, consisting of 10 randomly chosen source records from each of our 8 GeoNames types, for a total of 80. This fixed test set allows us to compare the performance of the rules and machine learning methods evaluated on the exact same data subset. We also use this fixed test set to plot a learning curve for different feature combinations, that is, to show how performance changes as we use more and more training data to train the random forest (results in section 5.3).

All the processing code was implemented in python, relying primarily on the *pandas*[4] library to work with data tables and compute matching features, and the *scikit-learn* library for machine learning (Pedregosa *et al*. 2011). We make our code and analysis files publicly available.[5]

4.3.2. Candidate selection

The first gazetteer matching step is candidate selection. For the training set, the aim of candidate selection is primarily to find (hard) negative record pairs to go alongside the known positive record pairs in order to train a successful classifier, and to make the subsequent machine learning independent from the absolute size of the data to be matched. For the test set however, candidate selection should aim to retain as many positive record pairs as possible (alongside some negative pairs), since the true positives would not be known in advance in a real matching scenario. Thus theoretically different candidate selection methods could be used for the training and test set, especially to try to increase recall on the test set, since any true positive pairs not retained during candidate selection will simply not make it to the classification step.

We selected candidate matches in the same way for both training and test pipelines. After a loose feature type filter (Table 2), we calculate the Levenshtein distance on names and the point-to-point distance on geometries, then combine these for an overall

score, similar to our *linear-combination* procedure. We then retain the top k candidates per source feature, where k was set experimentally by comparing performance for a range of values of k (results in section 5.2).

4.3.3. Feature extraction

The second matching step is to compute a range of matching features between all of the candidate source-target record pairs for use in the random forest. We chose a wide range of features to capture the similarity of the record names, locations, feature types, and geographical context via elevation and land cover. For land cover data, we use nationally-produced data for Switzerland containing a top level 6-class categorization of land cover.[6]

Our full list of features is as follows:

- **names**: for primary names, we calculate the Levenshtein distance, the normalized Levenshtein-Damerau distance, the Jaro similarity, and the Jaro-Winkler similarity; we additionally calculate the Levenshtein distance on any alternate names (present in GeoNames only) and on names with a comma, where we remove the comma and move the token following the comma to the beginning; finally we also take as a feature the minimum Levenshtein distance of those calculated.
- **geometries**: we calculate the point-to-point distance between gazetteer records (Vincenty 1975).
- **feature types**: we use one-hot encoding to encode feature type information, which removes any need to manually align our differing feature type hierarchies.
- **elevation**: we calculate the absolute difference between elevation values associated with the placenames in each gazetteer (essentially a measure of relief).
- **land cover**: we derive three land cover features from the land cover data. First, we find the land cover class of the nearest cell for both the source and target record, then one-hot encode this class. Second, we find the most frequent land cover class of the nearest 9 cells for each record and again one-hot encode this class. Finally, we calculate a feature we call 'land cover distance', where we take the counts of the 6 land cover classes in the 9 nearest cells for both source and target record (for example $[0, 4, 3, 1, 0, 1]$ and $[0, 4, 2, 0, 0, 3]$) then take the sum of the absolute value of the difference between these arrays (for example $[0, 0, 1, 1, 0, 2]$ for the absolute difference and $1 + 1 + 2 = 4$ for the sum, our final numeric feature).

4.3.4. Classification

We tested various combinations of the matching features we describe above, guided by our literature review. Based on reported strong performance of the Levenshtein distance for location name matching (Sehgal *et al*. 2006, McKenzie *et al*. 2014), we first formed a *basic* model using the minimum Levenshtein distance and the point-to-point distance. As a variant of the *basic* model, we formed a *str* model where we included all of our name matching features alongside point-to-point distance. We then added feature types to these two models, forming our *basic-type* and *str-type* models. We formed *str-elev-lc* to test whether we could compensate for a lack of feature type information by using elevation and land cover features. Finally, we tested 3 variants where most or all features are present, including feature types (*str-type-lcd, all-min, all*).

The feature combinations we focused on are thus as follows:

- **basic**: minimum Levenshtein distance and geographical distance.
- **str**: all name (string) features and geographical distance.
- **basic-type**: minimum Levenshtein distance, geographical distance, and encoded feature types.
- **str-type**: all name (string) features, geographical distance, and encoded feature types.
- **str-elev-lc**: all name (string) features, geographical distance, elevation, and all land cover features (no feature type information).
- **str-type-lcd**: all name features, geographical distance, encoded feature types, and land cover distance.
- **all-min**: minimalist version still using one feature per category: minimum Levenshtein distance, geographical distance, encoded feature types, elevation, and land cover distance.
- **all**: all features.

4.4. Evaluation

We evaluate the performance of both our rule-based and machine learning based matching using standard precision, recall, and F1 measures (Sehgal *et al*. 2006). Precision is defined as the number of positive matches correctly found divided by the total number of positive matches found, while recall is defined as the number of positive matches correctly found divided by the total number of positive matches that were to be found. Since precision can typically be optimized at the expense of recall and vice-versa, F1 is a measure combining precision and recall through their harmonic mean and thus summarizes the overall performance.

In our case, there is an additional complexity to be aware of with respect to recall. Calculating recall requires knowing how many positive matches there were in total involving the source records in the test set. Some of these positive matches will however not make it through the candidate selection stage, such as pairs with both dissimilar names and locations. Since in our test pipeline, we only keep candidate matches that were retained in candidate selection, we have two sets of matches we can use as the recall denominator: the full set of matches involving our test records (*overall recall*) or just those matches that made it to the classification stage (*classification recall*).

We consider *overall recall* to be the more meaningful recall of the two, since in a real-world scenario, the full set of correct matches for each source record is not known in advance, but instead the correct target records have to be found via candidate selection. The *classification recall* however serves as a useful evaluation of the classification stage specifically, without considering directly how well or poorly candidate selection performed. We can calculate an upper bound for overall recall right after candidate selection by looking at the percentage of positive matches that we retained out of the full set of known positives for the test records. We refer to this upper bound as *max recall* and present it alongside the other values described.

5. Results and interpretation

We describe here the results of our matching experiments, first presenting results from our rule-based matching procedures, then detailing our results using random forests. To describe the performance of our random forests, we first show how we fixed *k*, the number of target record candidates per source record used for candidate selection. We then present our main results: precision, recall, and F1 performance over 20 runs for all our combinations of matching features. Finally, we report results obtained on our fixed, feature-type balanced test set and plot a learning curve to show how performance changes as we increase the size of the training set.

5.1. Rule-based matching

Our rule-based matching results are shown in Table 3, in order of increasingly complex rules. Results are shown for a distance threshold of 5km for *name-threshold*, type-specific thresholds of either 5km or 15km for *name-custom-threshold* and *multi-threshold* (see Table 2), and additional thresholds of 400m of elevation difference and 8 units of land cover distance for *multi-threshold*. For *linear-combination*, results are shown using a single overall threshold and two weighting schemes for textual and geographical distance, one for lakes, streams, and valleys (LK, STR, VAL) which gives lesser weight to geographical distance, and one for all other types which gives equal weight to both.

The relatively high precision of *random-baseline*, where we chose a random exact name match as the match, shows that a significant proportion of the data can be dealt with using names alone. For two more complex sets of rules, *name-custom-threshold* and *linear-combination*, we were able to obtain good overall performance, with F1 values reaching around 0.85. Our *multi-threshold* approach, which employs additional thresholds on elevation and land cover, can clearly increase precision, but at the cost of much lower recall, and overall lower F1 values. Indeed, the best F1 performance we obtained with *multi-threshold* after trying a range of values for the additional thresholds was simply to not have these thresholds, which is equivalent to *name-custom-threshold*.

The higher performance of *name-custom-threshold* and *linear-combination* come at the cost of having to set several parameters in an ad hoc fashion, including multiple distance thresholds for *name-custom-threshold* and an overall score threshold with type-specific weightings for *linear-combination*. This threshold and the weightings could be optimized further, particularly to favour precision over recall or vice-versa, depending on the task requirements, by setting up a grid-search using one of the performance measures detailed above and evaluating the performance on a held-out or cross-validation sample. However, both trying to combine many matching features and trying

Table 3. Results for rule-based matching.

name of run	precision	recall	F1
random-baseline[a]	0.793	0.575	0.666
name-threshold	0.876	0.788	0.830
name-custom-threshold	0.843	0.861	0.852
multi-threshold	0.914	0.677	0.778
linear-combination	0.871	0.833	0.852

[a]random-baseline results were averaged over 10 runs.

to fully optimize rule-based matching leads us quickly down a data-driven path for which machine learning is better suited.

5.2. Machine learning based matching

We now present our machine learning based matching results, starting with how we fixed a value for *k*, the number of target records retained per source record during candidate selection. Figure 3 shows the overall performance (F1) of our machine learning pipeline for different feature combinations and values of *k*. Values of *k* below 10 appear to decrease F1, while a value of 30 appears to be optimal for most of the feature combinations tested and was used in the subsequent experiments.

In Figure 4, we present our main results for the full machine learning pipeline, showing the performance of different feature combinations over 20 runs with $k = 30$, in terms of F1, precision, (overall) recall, and classification recall. As mentioned, classification recall is calculated on just those positive record pairs retained during candidate selection, whereas overall recall is calculated using the full set of annotated positive pairs. For each run, all feature combinations were trained and tested on a particular random data split. This means each feature combination was evaluated against the same 20 sets of test records, presenting mixed feature type profiles and variations in difficulty.

Overall, median F1 values start at around 0.80 for our *basic* feature combination and increase up to 0.90 for the *str-type-lcd* combination. Visible on the F1 plot is a clear

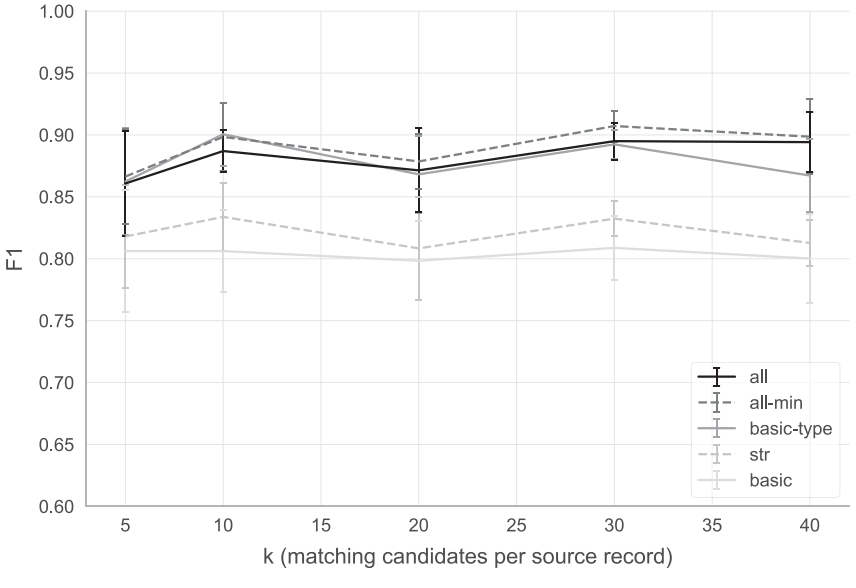


Figure 3. Mean and standard deviation of F1 vs *k* (number of target records retained per source record during candidate selection) over 5 runs per data point (tested over 5 values of *k* and 5 feature combinations). Feature combinations plotted: *basic*: minimum Levenshtein distance and geographical distance; *str*: all name (string) features and geographical distance; *basic-type*: minimum Levenshtein distance, geographical distance, and encoded feature types; *all-min*: minimum Levenshtein distance, geographical distance, encoded feature types, elevation, and land cover distance; *all*: all features.

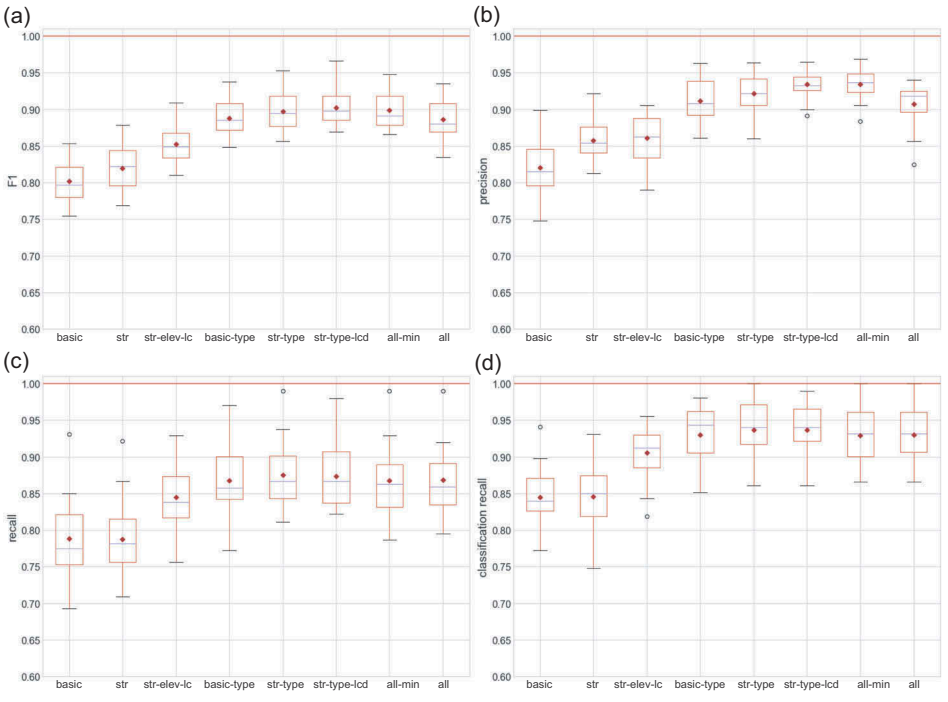


Figure 4. Box plot of medians (blue lines) with interquartile range and means (red diamonds) for: (a) F1 (b) precision (c) overall recall (d) classification recall vs. named combinations of matching features. Feature combinations: *basic*: minimum Levenshtein distance and geographical distance; *str*: all name (string) features and geographical distance; *basic-type*: minimum Levenshtein distance, geographical distance, and encoded feature types; *str-type*: all name (string) features, geographical distance, and encoded feature types; *str-elev-lc*: all name (string) features, geographical distance, elevation, and all land cover features (no feature types); *str-type-lcd*: all name features, geographical distance, encoded feature types, and land cover distance; *all-min*: minimum Levenshtein distance, geographical distance, encoded feature types, elevation, and land cover distance; *all*: all features.

difference between the feature combinations not using encoded feature types (*basic* and *str*) and those that do (the 5 rightmost on the plot), whose worst runs are all better than the median of the former two feature combinations. Somewhere in between we find the *str-elev-lc* combination, which lacks feature type information but uses all other matching features, offering F1 performance above the *basic* and *str* models, but below all models using encoded feature types. By examining the plots for precision and recall, we see that *str-elev-lc*'s performance advantage over *str* is almost entirely due to increased recall.

We obtained strong overall performance using the 5 combinations incorporating feature types as matching features, with no clear advantage of one over the others despite the addition of elevation and land cover features in some combinations (*all-min, all*) but not others (*str-type*). The *str-type-lcd* combination provides the best results over these 20 runs, with the highest median, mean, upper quartile, and lower quartile for F1. In Table 4 we present the mean values for these same runs, alongside the maximum overall recall (*max recall*) obtainable based on the record pairs retained at the candidate

Table 4. Mean values (over 20 runs) of precision, recall, and F1 for named feature combinations. Feature combinations: *basic*: minimum Levenshtein distance and geographical distance; *str*: all name (string) features and geographical distance; *basic-type*: minimum Levenshtein distance, geographical distance, and encoded feature types; *str-type*: all name (string) features, geographical distance, and encoded feature types; *str-elev-lc*: all name (string) features, geographical distance, elevation, and all land cover features (no feature types); *str-type-lcd*: all name features, geographical distance, encoded feature types, and land cover distance; *all-min*: minimum Levenshtein distance, geographical distance, encoded feature types, elevation, and land cover distance; *all*: all features.

feature combination	precision	recall	F1	max recall
basic	0.820	0.788	0.802	
str	0.857	0.788	0.820	
str-elev-lc	0.861	0.845	0.852	
basic-type	0.912	0.868	0.888	0.919
str-type	0.921	0.875	0.897	
str-type-lcd	0.934	0.874	0.902	
all-min	0.934	0.867	0.899	
all	0.907	0.868	0.886	

selection stage. Maximum recall changes with every run since the source records in the test set, and thus their candidate matching target records, also change, but this value is independent of the classification, and thus the mean is constant over all feature combinations.

5.3. Feature-type-balanced test set

In order to get further information about how our machine learning and rule-based matching perform, including how performance varies by feature type and how the machine learning methods respond to increasing amounts of training data, we ran tests using a fixed, feature-type balanced test set (as described in section 4.3.1). The results of these experiments are presented in Figure 5. Figure 5(a) shows the F1 performance on 5 representative feature types for a selection of matching strategies (3 rule-based and 4 machine learning based methods, prefixed by 'rf-' for random forests). This breakdown by type and matching method shows a generally more balanced performance across feature types for the machine learning strategies, particularly those that make use of encoded types (*basic-type, str-type, str-type-lcd*). It is also clear that much of the F1 performance gain of the higher performing strategies comes from doing much better on the worst performing type, streams (STM), while retaining strong performance (F1 >0.8) on the other types. Finally, the rule-based method that considers types via type-specific thresholds (*name-custom-threshold*) is doing better on all the plotted types than the machine learning method which does not consider types (*str*).

Figure 5(b) shows the F1 performance of our machine learning pipeline on the feature-type balanced test set as we increase the size of the training set by incrementally adding 40 randomly chosen source records in a step-wise fashion. Here, the 3 better performing feature combinations show continued improvement with increasing training data, but the *basic* model does not and instead seems to plateau when the training

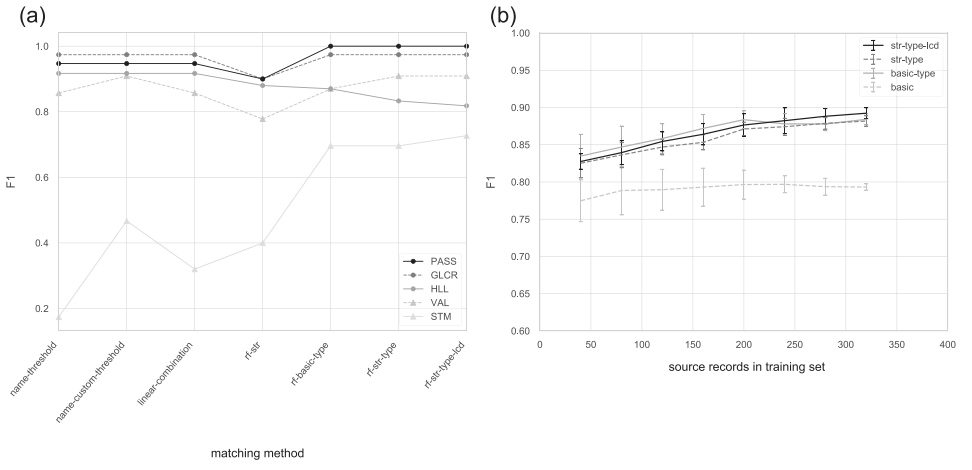


Figure 5. F1 performance according to (a) the matching strategy used (3 rule-based from the left and 4 machine learning based methods from the right, prefixed by *rf-*) broken down by feature type and (b) the number of source records used in the machine learning training pipeline, showing the mean and standard deviation over 10 runs using incrementally more randomly chosen records. Feature combinations plotted: *basic*: minimum Levenshtein distance and geographical distance; *basic-type*: minimum Levenshtein distance, geographical distance, and encoded feature types; *str-type*: all name (string) features, geographical distance, and encoded feature types; *str-type-lcd*: all name features, geographical distance, encoded feature types, and land cover distance.

pipeline uses around 200 source records. In general this means there is potential for our random forests to perform even better than they are, were there more annotated data available for training, above using the full set of 320 source records that are not in the feature-type balanced test set.

6. Discussion

Gazetteer matching is an important, real-world problem, where the very large numbers of records involved mean that small differences in precision or recall can have large implications. In this work, we performed cross-gazetteer matching on a set of natural feature records by implementing rule-based methods and machine learning methods using random forests. Our rule-based methods gave good results (Table 3), but our best machine learning models offered an F1 increase of 6% over the best rule-based results. However, rule-based and machine learning performance was similar when considering only record names and locations. Once feature types were incorporated as matching features in random forests, our models all achieved mean F1 values above 0.88.

This importance of gazetteer feature types on matching performance has received surprisingly limited research attention, with most previous work focusing on a very narrow set of types such as POIs or cities (Zheng *et al*. 2010, Martins 2011, Dalvi *et al*. 2014, McKenzie *et al*. 2014). Despite little guidance on how to effectively handle feature types, doing so was crucial for the natural feature records treated in this work. Thanks to one-hot encoding for categorical variables, incorporating feature types into random forests was simple and enabled the random forests to adapt to types, from multiple

type hierarchies, in a data-driven fashion. In contrast, considering types in rule-based processing was less straightforward, requiring semantic knowledge of the types (e.g. for type-specific distance thresholds), potentially manually aligning type hierarchies (as in Hastings 2008, Morana *et al*. 2014), and tailoring decisions to our particular datasets.

Random forests also offered a more balanced performance across feature types compared to rules, arguably as a result of this flexible approach to types. Though to our knowledge no direct comparison of rules and machine learning has been performed on matching geographical data, a similar finding was obtained in a work on deduplicating a dataset of inventors. Indeed, Ventura *et al*. (2015) compared the performance of rule-based methods against supervised learning using random forests and found that random forests offered much more robust performance with respect to data subsets with varying characteristics. In addition to robustness to feature type profiles, our experiments show that random forests perform increasingly well with more training data, suggesting that extra gains in F1 can be obtained through further annotation. Similar experiments were performed by Zheng *et al*. (2010), who varied the size of their training and testing sets together, and found that overall accuracy increased with the dataset size. Despite these advantages of supervised learning over rules, the performance gains came with costs, including greater complexity and more person-hours spent on implementing a random forest pipeline than on rule-based matching. Since simple rules performed well for the subset of data which was relatively easy (exact or near-exact name matches and very short point-to-point distances), deciding on a matching strategy for a different dataset would require careful consideration of these trade-offs.

Returning to the issue of complexity, implementing a realistic machine learning pipeline required carefully thinking about, and experimentally verifying, processing decisions including how to select match candidates, how to prepare the test set, and what ratio of negatives to positives to use. We found no clear methodological consensus in the literature, and even considerable disagreements, such as on the issue of negative-to-positive pair ratios, with some using 1:1 ratios (Zheng *et al*. 2010, Martins 2011), and others up to 30:1 (Sehgal *et al*. 2006). However, our decision to closely mimic a real-world scenario, treating test records as if they were unannotated, influenced many downstream decisions. This meant letting the testing pipeline find all (positive) matches through candidate selection, and not adding our annotated matches to the test set. This in turn meant splitting source records at the very beginning of the pipeline and choosing a candidate selection method likely to return not just hard negatives, but positive matches – as many as possible to maximize recall. We thus selected candidates based on considering multiple matching features at once (similar to Zheng *et al*. 2010, Morana *et al*. 2014), avoiding techniques geared more specifically towards negative selection (Sehgal *et al*. 2006, Martins 2011) and which could potentially limit recall. Finally, choosing how many matching candidates to keep per source record (i.e. k) was a purely experimental decision, optimizing for F1 on the full pipeline. We found that low values of k (in other words, low ratios of negatives to positives, assuming an average positive match count around 1 per record) limited recall and settled on a k of 30, similar to Sehgal *et al*. (2006).

After this closer look at how our methodology compares to existing work, a related question is, how do our results compare with previous published results? From our survey of

Table 5. Data-related aspects of selected papers comparable to the present work.

Authors	Task	Data description	Feature types
Sehgal *et al.* 2006	Gazetteer matching	US & UK authoritative data for Afghanistan	Varied (all)
Hastings 2008	Gazetteer matching	3 datasets covering Lake Tahoe (US)	Administrative, cultural, and water
Smart *et al.* 2010	Gazetteer matching	UK data from: GeoNames, Ordnance Survey, OpenStreetMap, Yahoo! Where on Earth, Wikipedia	Varied (all)
Zheng *et al.* 2010	Deduplication	POIs and yellow page records for Beijing (China)	POIs
Martins 2011	Deduplication	Global, mixed source	Populated places
Gonçalves 2012	Deduplication	Global, mixed source	Populated places
McKenzie *et al.* 2014	Gazetteer matching	Foursquare, Yelp (US)	POIs
Dalvi *et al.* 2014	Deduplication	Facebook Places (US)	POIs

Table 6. Method-related aspects of selected papers comparable to the present work.

					Best performance		
Authors	Approach	Method summary	Annotated positive pairs	neg:pos ratio	p	r	f1
Sehgal *et al.* 2006	Machine learning	Logistic regression, voted perceptron, SVM	2,006	30:1	.96	.92	.94
Hastings 2008	Rule-based	Consider names, footprints, feature types	252	N/A	.89	.22	.35
Smart *et al.* 2010	Rule-based	Consider names and locations	16	N/A	1.0	.44	.61
Zheng *et al.* 2010	Machine learning	Decision tree with bootstrap aggregating	800	1:1	.89	.87	.88
Martins 2011	Machine learning	SVM, alternating decision tree	1,927	1:1	.99	.98	.98
Gonçalves 2012	Machine learning	SVM, alternating decision tree, random forest	4,401	1:1	.97	.97	.97
McKenzie *et al.* 2014	Regression	Weighted multi-attribute model	100	N/A	accuracy = .97		
Dalvi *et al.* 2014	Machine learning	Unsupervised language model using local context	4,000	7:2	.90	.90	.90

the literature, it is clear that the variety of datasets and methods used make comparison difficult. Nonetheless, we have compiled a list of the most comparable papers and present a structured summary of these, with data-related aspects in Table 5, and method-related aspects, including best reported performance, in Table 6. We list 'task' as a data-related aspect since the key difference between a deduplication task and a (cross-)gazetteer matching task lies in whether data are already structured according to a single schema and single feature type hierarchy. With a single type hierarchy, a range of additional matching features can be used, for instance type equality or the hierarchical distance between types (Zheng *et al.* 2010, Martins 2011).

Visible is the predominance of datasets featuring POIs and populated places, which could arguably simplify matching due to low feature type diversity and, especially for POIs, a predictable spatial granularity for the records. POIs however present their own challenges, including potentially very large dataset sizes (Zheng *et al.* 2010, Dalvi *et al.* 2014) and particular naming patterns, which Zheng *et al.* (2010) tackle using custom token-based name similarity metrics, and Dalvi *et al.* (2014) using an innovative unsupervised language modeling approach. These two works achieve results similar to ours, with F1 values reaching 0.88 (Zheng *et al.* 2010) and 0.90 (Dalvi *et al.* 2014). McKenzie

et al. (2014) also report on a dataset of POIs, developing a regression approach with a wide set of matching features, but their evaluation is mainly performed on an artificial test set where 100 exact matches were created. In this rather unrealistic testing scenario, they report an accuracy of 0.97, but mention performance decreases when random pairs are selected from data, reporting F-scores of 0.35 and 0.32. In Martins (2011) and a follow-up work (Gonçalves 2012), very high deduplication performance is reported, with F1 values reaching 0.98 and 0.97, respectively. However, the authors note that their dataset of coarse-grained records (populated places) may not be particularly challenging, with F1 values reaching 0.97 when using only matching features based on name similarity (Martins 2011).

Two other works (Hastings 2008, Smart *et al*. 2010) are broadly similar to each other in that they use rule-based processing and datasets with diverse feature types, but don't rigorously evaluate their approaches. Smart *et al*. (2010) manually examine only a very small number of individual cases (n = 16), achieving perfect precision, but a nominal recall of only 0.44, and Hastings (2008) present a descriptive evaluation, for which we calculated precision to have been 0.89, and recall only 0.22. In contrast, Sehgal *et al*. (2006) perform a rigorous quantitative evaluation of their machine learning based matching, reporting a range of values for precision, recall, F1, and accuracy, including how these change as a function of the ratio of negatives-to-positives. We perform comparable tests on setting a value of *k* (Figure 3), and complement this more extensive evaluation by performing multiple runs to test robustness to both individual training data sets and different training set sizes.

A final important point about our ability to compare our results to previous work is that it is at times unclear how matches are found for test records. If a single static collection of positive and negative record pairs is used to train and test a classifier, for example via cross-validation, then it is likely that the full set of annotated positives automatically finds it way into each testing fold (Figure 6). This would optimistically bias performance compared to a scenario in which matches for test records must be found via candidate selection, and from a large number of such candidate records, as is the case in our pipeline. In addition, adding random negatives into a test set will

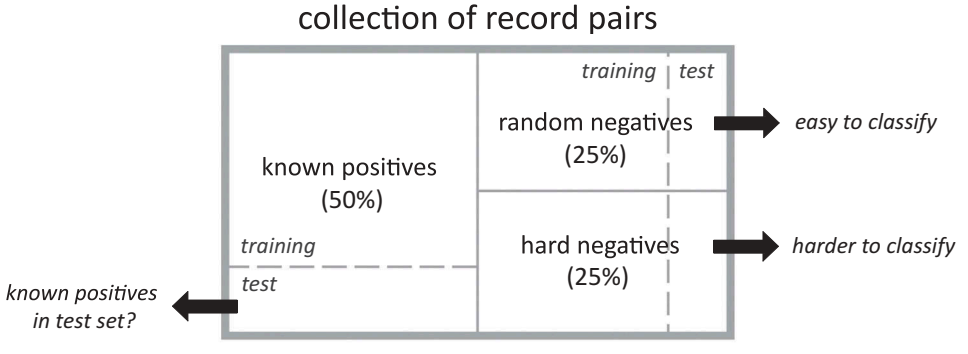


Figure 6. Potential test set composition when using cross-validation with a collection of record pairs built with a 1:1 negative to positive ratio and where 50% of negatives are chosen randomly.

optimistically bias accuracy, since these record pairs will overwhelmingly be highly dissimilar, and thus easy to classify as non-matches (Figure 6). Whether random negatives have any effect on precision and recall is unclear, but we opted against using them in our training pipeline since we already had quite varied training data, thanks in part to using a high value of *k*. Furthermore, random forests rely on random samples of the training data for each tree and thus naturally include variations from different samples of the distribution.

7. Conclusion

In this work, we tackled the problem of matching natural feature records across two gazetteers. We showed that good performance could be obtained using relatively simple rules, but that machine learning using random forests offered not only better performance, but greater flexibility, obviating the need to manually align feature types and tune thresholds. Random forests also offered a more balanced performance across feature types and the potential for even better performance by increasing the amount of training data through further annotation. We emphasize that creating a training dataset in order to implement a machine learning solution was both more straightforward than handcrafting rules for gazetteer matching and more easily generalizable. With random forests, the biggest performance increase over basic models using string similarity and geographical distance came from incorporating feature types as matching features, after which all tested models performed similarly well. However, all classifiers which included some representation of feature type, including by using land cover classes and elevation instead of gazetteer feature types, outperformed simpler string and geographical distance based classifications.

Although our results were obtained on the specific case of matching records between a Swiss national gazetteer and GeoNames, they have more general implications, particularly in the context of growing interest in spatial data science. We make the following recommendations for future work in this area:

- Gazetteer matching is influenced by feature types and therefore any processing decisions related to these feature types for matching should be explicitly described.
- Training and evaluation datasets should be carefully designed so as not to make classification problems unrealistically straightforward or difficult (e.g. nearby toponyms with similar names which are not matches should be explicitly included in test data). Future work should also consider the impact of candidate selection on overall performance.
- Since gazetteer data are often snapshots, and may also not be freely available, making at a minimum annotated data available will both increase the potential for reproducibility, and allow other researchers to understand the properties of the snapshots investigated. Use of shareable markdown (e.g. R-Markdown, Juypter notebooks) will further increase reproducibility and make all stages of processing more transparent.

Notes

1. The term 'feature' is both widely used in the GIScience community to refer to geographical entities and in the machine learning community to refer to properties of the data used to

train models. To avoid potential ambiguity, we refer to geographical features as 'entities' in a real-world context or 'records' in a gazetteer context, and we use 'features' to refer to 'matching features' in the machine learning sense. We however maintain the use of the widely used two-word expressions 'feature type' to refer to the catalogued type of geographical entities and 'natural features' to refer to the subset of geographical entities which are not human-made.
2. https://shop.swisstopo.admin.ch/en/products/landscape/names3D.
3. CH.zip from http://download.geonames.org/export/dump/.
4. https://pandas.pydata.org/.
5. https://github.com/eacheson/machine-learning-gazetteer-matching.
6. https://www.bfs.admin.ch/bfs/de/home/dienstleistungen/geostat/geodaten-bundesstatistik/boden-nutzung-bedeckung-eignung/arealstatistik-schweiz/bodenbedeckung.html.

Acknowledgments

RSP gratefully acknowledges support from Swiss National Science Foundation Project EVA (166788).

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Schweizerischer Nationalfonds zur Förderung der Wissenschaftlichen Forschung [166788].

Notes on contributors

Elise Acheson is a PhD student in the Geocomputation Unit at the University of Zurich. Her work focuses on automatically extracting geographical information from various textual sources.

Michele Volpi is a Senior Data Scientist at the Swiss Data Science Center, a joint venture between ETH Zurich and EPFL aiming at accelerating the adoption of data science within the ETH Domain and the Swiss academic community at large. His main research activities are at the interface of computer vision, machine learning, and deep learning, focusing on the extraction of information from aerial and satellite imagery and from geospatial and environmental data in general.

Ross S. Purves is a Professor at the Department of Geography of the University of Zurich. His research interests focus on how we can answer and explore societally relevant geographic questions paying attention to vagueness and uncertainty, often using unstructured data in the form of text as a primary source.

References

Acheson, E., *et al*., 2017b. Gazetteer matching for natural features in switzerland. *In*: Christopher B. Jones and Ross S. Purves, eds. *Proceedings of the 11th Workshop on Geographic Information Retrieval, GIR'17* New York, NY, USA: ACM, 11:1–11: 2.

Acheson, E., De Sabbata, S., and Purves, R.S., 2017a. A quantitative analysis of global gazetteers: patterns of coverage for common feature types. *Computers, Environment and Urban Systems*, 64, 309–320. doi:10.1016/j.compenvurbsys.2017.03.007

Adams, B., McKenzie, G., and Gahegan, M., 2015. Frankenplace: interactive thematic mapping for Ad Hoc exploratory search. *In*: Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi, eds. *Proceedings of the 24th International Conference on World Wide Web, WWW '15*, Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 12–22.

Ahlers, D., 2013. Assessment of the accuracy of GeoNames gazetteer data. *In*: Christopher B. Jones and Ross S. Purves, eds. *Proceedings of the 7th Workshop on Geographic Information Retrieval, GIR '13*, New York, NY, USA: ACM, 74–81.

Berman, M.L., Mostern, R., and Southall, H., eds.. 2016. *Placing names: enriching and integrating gazetteers*. Bloomington, Indiana: Indiana University Press.

Brauner, D.F., Casanova, M.A., and Milidiú, R.L., 2007. Towards gazetteer integration through an instance-based thesauri mapping approach. *In*: C.A.D. Jr and A.M.V. Monteiro, eds. *Advances in geoinformatics*. Campos do Jordão (SP), Brazil: Springer Berlin Heidelberg, 235–245.

Breiman, L., 2001. Random forests. *Machine Learning*, 45 (1), 5–32. doi:10.1023/A:1010933404324

Brunner, T.J. and Purves, R.S., 2008. Spatial autocorrelation and toponym ambiguity. *In*: Chris Jones and Ross Purves, eds. *Proceedings of the 5th Workshop on Geographic Information Retrieval, GIR '08* New York, NY, USA: ACM, 25–26.

Christen, P., 2012. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Transactions on Knowledge and Data Engineering*, 24 (9), 1537–1555. doi:10.1109/TKDE.2011.127

Cooper, D. and Gregory, I.N., 2011. Mapping the English lake district: a literary GIS. *Transactions of the Institute of British Geographers*, 36 (1), 89–108. doi:10.1111/tran.2010.36.issue-1

Costa, G., 2011. Data de-duplication: a review. *In*: J. Kacprzyk, *et al*., eds. *Learning structure and schemas from documents*, Vol. 375. Berlin, Heidelberg: Springer Berlin Heidelberg, 385–412.

Dalvi, N., *et al*., 2014. Deduplicating a places database. *In*: Chin-Wan Chung, Andrei Broder, Kyuseok Shim, and Torsten Suel, eds. *Proceedings of the 23rd International Conference on World Wide* Web, WWW '14, New York, NY, USA: ACM, 409–418.

Elmagarmid, A.K., Ipeirotis, P.G., and Verykios, V.S., 2007. Duplicate record detection: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 19 (1), 1–16. doi:10.1109/TKDE.2007.250581

Fu, G., Jones, C.B., and Abdelmoty, A.I., 2005. Building a geographical ontology for intelligent spatial search on the web. *In*: M.H. Hamza, ed. *Proceedings of IASTED International Conference on Databases and Applications (DBA-2005)*, February. Innsbruck, Austria: ACTA Press, 167–172.

Gao, S., *et al*. 2017. Constructing gazetteers from volunteered big geo-data based on Hadoop. *Computers, Environment and Urban Systems*, 61 (Part B), 172–186. doi:10.1016/j.compenvurbsys.2014.02.004

Gelernter, J., *et al*., 2013. Automatic gazetteer enrichment with user-geocoded data. *In*: Dieter Pfoser and Agnès Voisard, eds. *Proceedings of the Second ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Infor*mation, GEOCROWD '13, New York, NY, USA: ACM, 87–94.

Gonçalves, N.F.A., 2012. Gazetteer record linkage. Master's thesis. Lisbon: Instituto Superior Técnico. doi:10.1094/PDIS-11-11-0999-PDN

Hastings, J.T., 2008. Automated conflation of digital gazetteer data. *International Journal of Geographical Information Science*, 22 (10), 1109–1127. doi:10.1080/13658810701851453

Hill, L.L., 2006. *Georeferencing: the geographic associations of information*. Cambridge, MA: The MIT Press.

Janowicz, K. and Keßler, C., 2008. The role of ontology in improving gazetteer interaction. *International Journal of Geographical Information Science*, 22 (10), 1129–1157. doi:10.1080/13658810701851461

Keßler, C., *et al*. 2009. Bottom-up gazetteers: learning from the implicit semantics of geotags. *In*: K. Janowicz, M. Raubal, and S. Levashkin, eds. *GeoSpatial Semantics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 83–102.

Levenshtein, V.I., 1966. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii nauk SSSR*, 163 (4), 845–848.

Lieberman, M.D., Samet, H., and Sankaranayananan, J., 2010. Geotagging: using proximity, sibling, and prominence clues to understand comma groups. *In*: Ross Purves, Paul Clough, and Chris

Jones, ed. *Proceedings of the 6th Workshop on Geographic Information Retrieval, GIR '10*, New York, NY, USA: ACM, 6: 1–6: 8.

Martins, B., 2011. A supervised machine learning approach for duplicate detection over gazetteer records. *In*: Christophe Claramunt, Sergei Levashkin, and Michela Bertolotto, eds. *GeoSpatial semantics*, Lecture notes in computer science, May. Berlin, Heidelberg: Springer, 34–51.

McKenzie, G., Janowicz, K., and Adams, B., 2014. A weighted multi-attribute method for matching user-generated points of interest. *Cartography and Geographic Information Science*, 41 (2), 125–137. doi:10.1080/15230406.2014.880327

Morana, A., *et al*., 2014. GeoBench: a geospatial integration tool for building a spatial entity matching benchmark. *In*: Yan Huang, Markus Schneider, Michael Gertz, John Krumm, and Jagan Sankaranarayanan, eds. *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '14, New York, NY, USA: ACM, 533–536.

Murphy, A.H., 1996. The finley affair: a signal event in the history of forecast verification. *Weather and Forecasting*, 11 (1), 3–20. doi:10.1175/1520-0434(1996)011<0003:TFAASE>2.0.CO;2

Olteanu, A., Musti'ere, S., and Ruas, A., 2006. Matching imperfect spatial data. *In*: M. Caetano and M. Painho, eds. *Proceedings of 7th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences*, July. Lisbon, Portugal, 7–9.

Pedregosa, F., *et al*. 2011. Scikit-learn: machine learning in python. *Journal of Machine Learning Research*, 12, 2825–2830.

Popescu, A., Grefenstette, G., and Mo¨Ellic, P.A., 2008. Gazetiki: automatic creation of a geographical gazetteer. *In*: Ronald Larsen, Andreas Paepcke, José Borbinha, and Mor Naaman, eds. *Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '08*, New York, NY, USA: ACM, 85–93.

Purves, R.S., *et al*. 2007. The design and implementation of SPIRIT: a spatially aware search engine for information retrieval on the internet. *International Journal of Geographical Information Science*, 21 (7), 717–745. doi:10.1080/13658810601169840

Santos, R., *et al*., 2018. Toponym matching through deep neural networks. *International Journal of Geographical Information Science*, Taylor & Francis, 32 (2), 324–348. doi:10.1080/13658816.2017.1390119.

Sehgal, V., Getoor, L., and Viechnicki, P.D., 2006. Entity resolution in geospatial data integration. *In*: Rolf A. de By, and Silvia Nittel, eds. *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems, GIS '06*, New York, NY, USA: ACM, 83–90.

Smart, P.D., Jones, C.B., and Twaroch, F.A., 2010. Multi-source toponym data integration and mediation for a meta-gazetteer service. *In*: Sara Irina Fabrikant, Tumasch Reichenbacher, Marc van Kreveld, and Christoph Schlieder, eds. *Geographic information science*, Lecture notes in computer science, September, Berlin, Heidelberg: Springer, 234–248.

Smith, B. and Mark, D., 2003. Do mountains exist? Towards an ontology of landforms. *Environment and Planning B (Planning and Design)*, 30 (3), 411–427. doi:10.1068/b12821

Ventura, S.L., Nugent, R., and Fuchs, E.R.H., 2015. Seeing the non-stars: (Some) sources of bias in past disambiguation approaches and a new public tool leveraging labeled records. *Research Policy*, 44 (9), 1672–1701. doi:10.1016/j.respol.2014.12.010

Vincenty, T., 1975. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*, 23 (176), 88–93. doi:10.1179/sre.1975.23.176.88

Walter, V. and Fritsch, D., 1999. Matching spatial data sets: a statistical approach. *International Journal of Geographical Information Science*, 13 (5), 445–473. doi:10.1080/136588199241157

Zheng, Y., *et al*., 2010. Detecting nearly duplicated records in location datasets. *In*: Divyakant Agrawal, Pusheng Zhang, Amr El Abbadi, and Mohamed Mokbel, eds. *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '10*, New York, NY, USA: ACM, 137–143.

Zhu, R., *et al*., 2016. Spatial signatures for geographic feature types: examining gazetteer ontologies using spatial statistics. *Transactions in GIS*. doi:10.1111/tgis.2016.20.issue-3